Functional Dependencies

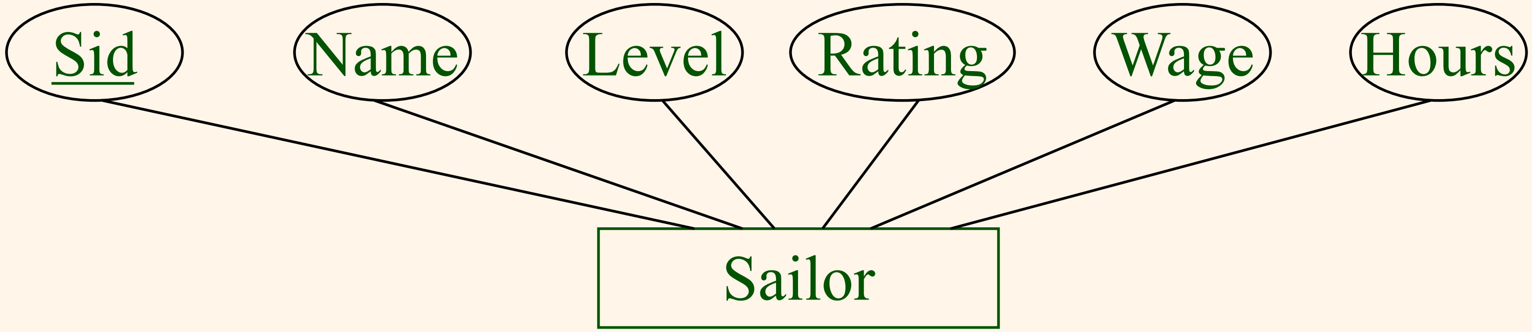
Last time

- Entity-Relationship diagrams as a modeling tool
- Translating ER diagrams to SQL

Today

- More theoretical material
  - Constraints
  - Redundancy
- Maier's online textbook Ch. 4-6
  - web.cecs.pdx.edu/~maier/TheoryBook/MAIER/

Entity-Relationship Diagram



Data Redundancy

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

- Application constraint: all sailors with the same rating have the same wage
- Problems due to data redundancy?

Problems due to Data Redundancy

- Problems:
  - *Update anomaly*: Can change W in just the first tuple of the relation, without corresponding changes to others
  - *Insertion anomaly*: What if we want to insert an employee and don’t know the hourly wage for their rating?
  - *Deletion anomaly*: If we delete all employees with rating 5, we lose the information about the wage for rating 5!
- Solution?

Relation Decomposition

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

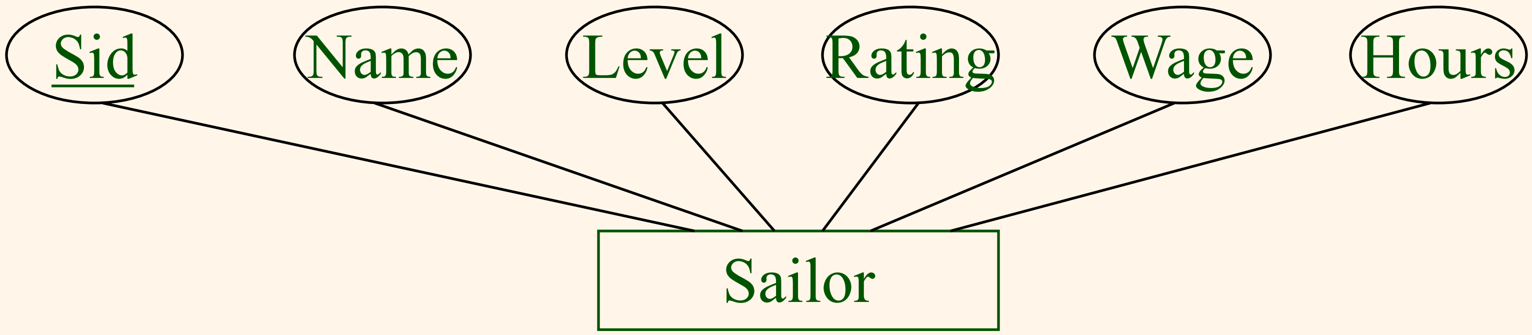
S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

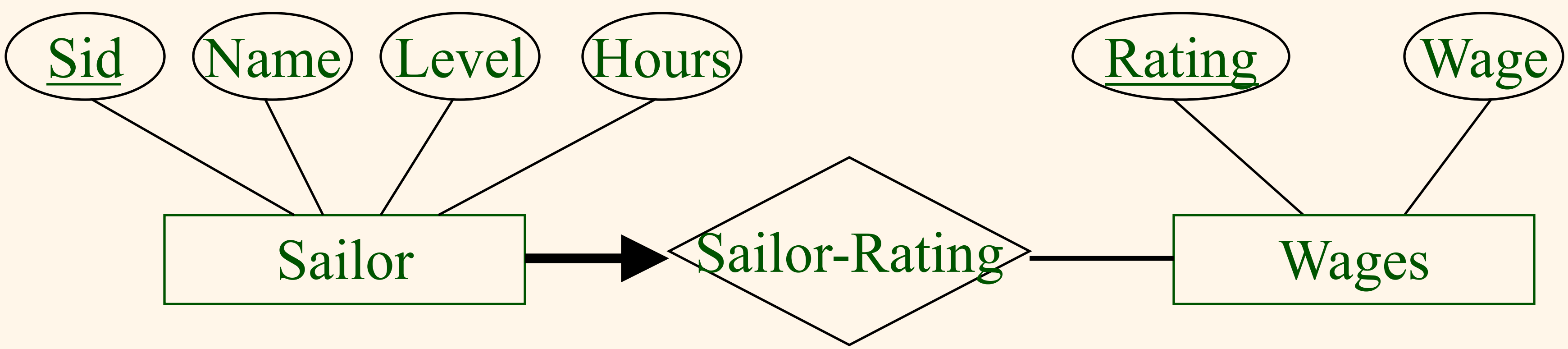
Wages

R	W
8	10
5	7

Problem?

Modifying ER Diagram





Decomposition

- Decomposing removes redundancy
- But some queries now require a join
- Some questions:
  - How do we detect redundancy?
  - Can decompositions cause other problems beyond performance?
  - Is there always one correct way to decompose?

Functional Dependencies (FDs)

- Help us understand redundancy
- Common real-world constraint: value of attribute A uniquely determines the value of attribute B
- Example:
  - Rating determines hourly wage
  - Beer name determines manufacturer name
- If two tuples agree on the A value, they also agree on the B value

Functional Dependencies (FDs)

- Suppose X and Y are sets of attributes,
- Functional dependency X→Y holds over R if:

$$\forall t \in R, s \in R, \pi_X(t) = \pi_X(s) \Rightarrow \pi_Y(t) = \pi_Y(s)$$

  - given two tuples in R, if their X values agree, then the Y values must also agree

- In our example: S.rating→ S.wage
- Relationship between keys and FDs?

Defining keys formally

- If X is a key for R, then R cannot contain two different tuples that have the same value of X

$$\neg\exists t, s \in R\ (t \neq s \wedge \pi_X(t) = \pi_X(s))$$

- Or equivalently

$$\forall t, s \in R, \pi_X(t) = \pi_X(s) \Rightarrow t = s$$

- In addition, no subset of X can have the above property (minimality of a *key* vs *superkey*)

FDs and Keys

- Letting U be the set of all attributes of R, we know that in the set relational model

$$\forall t, s \in R, \pi_U(t) = \pi_U(s) \iff t = s$$

FDs and Keys

- Putting it all together: if X is a key then

$$\forall t, s \in R, \pi_X(t) = \pi_X(s) \Rightarrow t = s$$

- But given that

$$\forall t, s \in R, \pi_U(t) = \pi_U(s) \iff t = s$$

- It follows that

$$\forall t \in R, s \in R, \pi_X(t) = \pi_X(s) \Rightarrow \pi_U(t) = \pi_U(s)$$

- I.e. $X \rightarrow U$, so there is a FD from X to all the attributes of R

FDs and redundancy

- Intuitively, if X→Y and we are storing the value of Y twice for the same value of X, we have redundancy

bid	color	type	rental_rate
1	red	slow	$100
2	blue	slow	$100
3	red	fast	$200
4	red	fast	$200
5	blue	fast	$200

Detecting redundancy

- Slightly more formally, need to:

- Find all the FDs $X \rightarrow Y$ that hold
- For each one, check if X is a key
  - Remember – many keys, not just primary key!
    - Netid/student id/ssn all keys for CU students…
- If it is not, we have found redundancy!

A few things we need

- How do we find all the FDs that hold?
- Some we know from domain knowledge (the external world)
  - E.g. time,room → coursenum
- Some we can infer using logic rules
  - If time,room→coursenum and coursenum→instructor, then time,room→instructor

Finding FDs – caution!

A	B	C
aaa	bbb	ccc
zzz	xxx	ddd
zzz	xxx	eee

- Can we be sure that this satisfies A $\rightarrow$ B ?
- No violations, sure…
- But maybe just got lucky
- E.g. address$\rightarrow$ lastname
  - Could well have a table where this is true
  - But not true in general!

Reasoning about FDs

- A set of FDs F **implies** an additional FD $X \rightarrow Y$ if on any relation where all FDs in F hold, $X \rightarrow Y$ also holds.
- We use the notation

$$F \models X \rightarrow Y$$

FD Implication

- For example:

$$\{A \to B, B \to C\} \models A \to C$$

- But:

$$\{A \to B, C \to B\} \not\models A \to C$$

- We can construct a relation where the first two hold but the third one doesn't

FD Implication

- How to find all FDs that are **implied** by an initial set F?
- Want some algorithm to **derive** them all automatically, with a program
  - That way we can be sure we haven't missed any

Armstrong's Axioms

- Let X, Y, Z be attribute sets
- **Reflexivity**: if $Y \subseteq X$ then $X \to Y$
- **Augmentation**: if $X \to Y$ then $XZ \to YZ$ for any Z
- **Transitivity**: if $X \to Y$ and $Y \to Z$ then $X \to Z$

Derivability

- If we can start with a set F and apply Armstrong's Axioms to obtain a new FD $X \to Y$ then F **derives** $X \to Y$
- Notation:

$$F \vdash X \to Y$$

- Example:

$$\{A \to B, B \to C\} \vdash A \to C$$

An example

- Let's do an example derivation
- Example: Contracts(*cid,sid,jid,did,pid,qty,value*), and:
  - C is the key: C $\rightarrow$ CSJDPQV
  - Project purchases each part using single contract: JP $\rightarrow$ C
  - Dept purchases at most one part from a supplier: SD $\rightarrow$ P
- Can you infer SDJ $\rightarrow$ CSJDPQV ?

Implication vs derivation

- Very important distinction:
- **Implication =** true in reality
- **Derivation =** can be computed automatically using a program

- These two don't need to coincide!

Implication vs derivation

- Maybe the axioms are "insufficient" and the program won't derive all the FDs that are implied by F!
- Maybe the axioms are wrong, eg. "if A -> B then B ->A" and the program will derive FDs that are not implied by the starting set!

Armstrong's Axioms

- They are sound
- If

$$F \vdash X \to Y$$

- Then

$$F \models X \to Y$$

- If we can derive an FD from F (mechanically), then it is actually implied by F
  - No "garbage axioms"

Let's do a proof

- Let's prove soundness of the transitivity axiom!
- Axiom says:

$$\{X \to Y, Y \to Z\} \vdash X \to Z$$

- To show soundness we need to show that for arbitrary X, Y and Z,

$$\{X \to Y, Y \to Z\} \models X \to Z$$

Soundess of transitivity

- Need to show that if a relation satisfies $X \rightarrow Y$ and $Y \rightarrow Z$, also satisfies $X \rightarrow Z$
- Let's proceed by contradiction: suppose it satisfies the first two but not $X \rightarrow Z$
- Then it must contain two tuples that agree on X but not on Z (picture is simplified, X, Y and Z are sets of attributes in general)

X	Y	Z
x1	???	z1
x1	???	z2

Soundess of transitivity

- Now let's fill in the table given what we know about the relation
- Satisfies $X \rightarrow Y$ so tuples must agree on Y

X	Y	Z
x1	y1	z1
x1	y1	z2

- Satisfies $Y \rightarrow Z$ so tuples must agree on Z
  - But we assumed they didn't
  - So we have a contradiction as desired.

Armstrong's Axioms

- They are sound
- If

$$F \vdash X \to Y$$

- Then

$$F \models X \to Y$$

- If we can derive an FD from F (mechanically), then it is actually implied by F
  - Axioms don't add "garbage" FDs

Armstrong's Axioms

- They are complete
- If

$$F \models X \to Y$$

- Then

$$F \vdash X \to Y$$

- Applying the axioms to F allows us to find **all** the "extra" FDs implied by F

Closures

- Starting set of FDs F
- The **closure** F^+ of F is the set of all FDs implied by FDs in F

$$F^+ = \{D \mid F \models D\}$$

- Can be computed using Armstrong's axioms
- Can be very large

Attribute closures

- Typically, we just want to check if a specific FD is in the closure of a set of FDs *F*.
- E.g. suppose

$$F = \{A \to D, AB \to E, BI \to E, CD \to I, E \to C\}$$

- And I want to know: does F imply $AE \to D$?

Attribute closures

- Want to see whether
$$F \models AE \rightarrow D$$
- No need to compute entire closure F^+
- Will instead compute the **attribute closure**
- Denote this as $(AE)^+$
- This is the set of all attributes K such that
$$(AE \rightarrow K) \in F^+$$
- Then can just check if $D \in (AE)^+$

Computing attribute closure of X

- Closure = X
- Repeat until no change:
  - For every $U \rightarrow V$ such that $U \subseteq$ closure
    - closure = closure $\cup V$

- Let's try this on our example, compute $(AE)^+$

$$F = \{A \rightarrow D, AB \rightarrow E, BI \rightarrow E, CD \rightarrow I, E \rightarrow C\}$$

Complexity of attribute closure algo?

- a = number of attributes
- f = number of FDs in F
- Each iteration of loop takes $O(af)$ time
- Outer loop executed at most f times
- Total: $O(af^2)$ time
- Can improve on this to a linear-time algorithm
  - See D. Maier's online textbook